CAS实现SSO单点登录原理

2013-04-08 | 所属分类：Cas SSO 原理

1. CAS 简介

1.1. What is CAS ?

CAS （ Central Authentication Service ） 是 Yale 大学发起的一个企业级的、开源的项目，旨在为 Web 应用系统提供一种可靠的单点登录解决方法（属于 Web SSO ）。

CAS 开始于 2001 年， 并在 2004 年 12 月正式成为 JA-SIG 的一个项目。

1.2. 主要特性

1、 开源的、多协议的 SSO 解决方案； Protocols： Custom Protocol 、 CAS 、 OAuth 、 Open ID 、 RESTful API 、 SAML1.1 、 SAML2.0 等。

2、 支持多种认证机制： Active Directory 、 JAAS 、 JDBC 、 LDAP 、 X.509 Certificates 等；

3、 安全策略：使用票据（ Ticket ）来实现支持的认证协议；

4、 支持授权：可以决定哪些服务可以请求和验证服务票据（ Service Ticket ）；

5、 提供高可用性：通过把认证过的状态数据存储在 TicketRegistry 组件中，这些组件有很多支持分布式环境的实现，如： BerkleyDB 、 Default 、 EhcacheTicketRegistry 、 JDBCTicketRegistry 、 JBOSS TreeCache 、 JpaTicketRegistry 、 MemcacheTicketRegistry 等；

6、 支持多种客户端： Java 、 .Net 、 PHP 、 Perl 、 Apache, uPortal 等。

2. SSO 单点登录原理

本文内容主要针对 Web SSO 。

2.1. 什么是SSO

单点登录（ Single Sign-On , 简称 SSO ）是目前比较流行的服务于企业业务整合的解决方案之一， SSO 使得在多个应用系统中，用户只需要 登录一次 就可以访问所有相互信任的应用系统。

2.2. SSO 原理

2.2.1. SSO 体系中的角色

一般 SSO 体系主要角色有三种：

1、 User （多个）

2、 Web 应用（多个）

3、 SSO 认证中心（ 1 个 ）

2.2.2. SSO 实现模式的原则

SSO 实现模式一般包括以下三个原则：

1、 所有的认证登录都在 SSO 认证中心进行；

2、 SSO 认证中心通过一些方法来告诉 Web 应用当前访问用户究竟是不是已通过认证的用户；

3、 SSO 认证中心和所有的 Web 应用建立一种信任关系，也就是说 web 应用必须信任认证中心。（单点信任）

2.2.3. SSO 主要实现方式

相关文章推荐

CAS实现SSO单点登录原理
CAS Ticket票据:TGT、ST、PGT、PT、PGTIOU
CAS的PGTURL、PGTIOU有什么作用?
CAS Spring Security 3 整合配置
cas sso登录流程httpwatch解析
cas代理(proxy)模式的原理及配置
SSO实现机制:Cookie机制和Session机制
CAS取消https方法配置
CAS Logout 注销
CAS、Spring Security、Ldap配置整合

SSO 的主要实现方式有：

1、 共享 cookies

基于共享同域的 cookie 是 Web 刚开始阶段时使用的一种方式，它利用浏览同域名之间自动传递 cookies 机制，实现两个域名之间系统令牌传递问题；另外，关于跨域问题，虽然 cookies 本身不跨域，但可以利用它实现跨域的 SSO 。如：代理、暴露 SSO 令牌值等。

缺点：不灵活而且有不少安全隐患，已经被抛弃。

2、 Broker-based(基于经纪人)

这种技术的特点就是，有一个集中的认证和用户帐号管理的服务器。经纪人给被用于进一步请求的电子身份存取。中央数据库的使用减少了管理的代价，并为认证提供一个公共和独立的 " 第三方 " 。例如 Kerberos 、 Sesame 、 IBM KryptoKnight （凭证库思想) 等。 Kerberos 是由麻省理工大学发明的安全认证服务，已经被 UNIX 和 Windows 作为默认的安全认证服务集成进操作系统。

3、 Agent-based （基于代理人）

在这种解决方案中，有一个自动地为不同的应用程序认证用户身份的代理程序。这个代理程序需要设计有不同的功能。比如，它可以使用口令表或加密密钥来自动地将认证的负担从用户移开。代理人被放在服务器上面，在服务器的认证系统和客户端认证方法之间充当一个 " 翻译 " 。例如 SSH 等。

4、 Token-based

例如 SecureID,WebID ，现在被广泛使用的口令认证，比如 FTP 、邮件服务器的登录认证，这是一种简单易用的方式，实现一个口令在多种应用当中使用。

5、 基于网关

6、 基于 SAML

SAML(Security Assertion Markup Language ，安全断言标记语言）的出现大大简化了 SSO ，并被 OASIS 批准为 SSO **的执行标准** 。开源组织 OpenSAML 实现了 SAML 规范。

3. CAS 的基本原理

3.1. 结构体系

从结构体系看， CAS 包括两部分： CAS Server 和 CAS Client 。

3.1.1. CAS Server

CAS Server 负责完成对用户的认证工作，需要独立部署，CAS Server 会处理用户名 / 密码等凭证 (Credentials) 。

3.1.2. CAS Client

负责处理对客户端受保护资源的访问请求，需要对请求方进行身份认证时，重定向到 CAS Server 进行认证。（原则上，客户端应用不再接受任何的用户名密码等 Credentials ）。

CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护受保护的资源。

3.2. CAS 原理和协议

3.2.1. 基础模式

基础模式 SSO 访问流程主要有以下步骤：

1. 访问服务： SSO 客户端发送请求访问应用系统提供的服务资源。

2. 定向认证： SSO 客户端会重定向用户请求到 SSO 服务器。

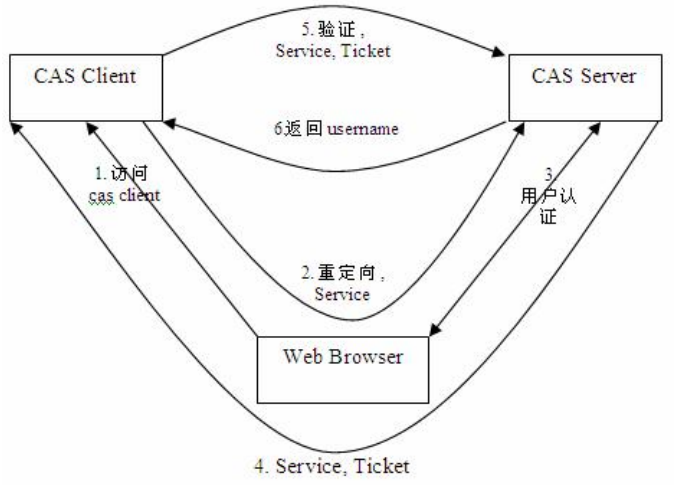
3. 用户认证：用户身份认证。

4. 发放票据： SSO 服务器会产生一个随机的 Service Ticket 。

5. 验证票据： SSO 服务器验证票据 Service Ticket 的合法性，验证通过后，允许客户端访问服务。

6. 传输用户信息： SSO 服务器验证票据通过后，传输用户认证结果信息给客户端。

下面是 CAS 最基本的协议过程：

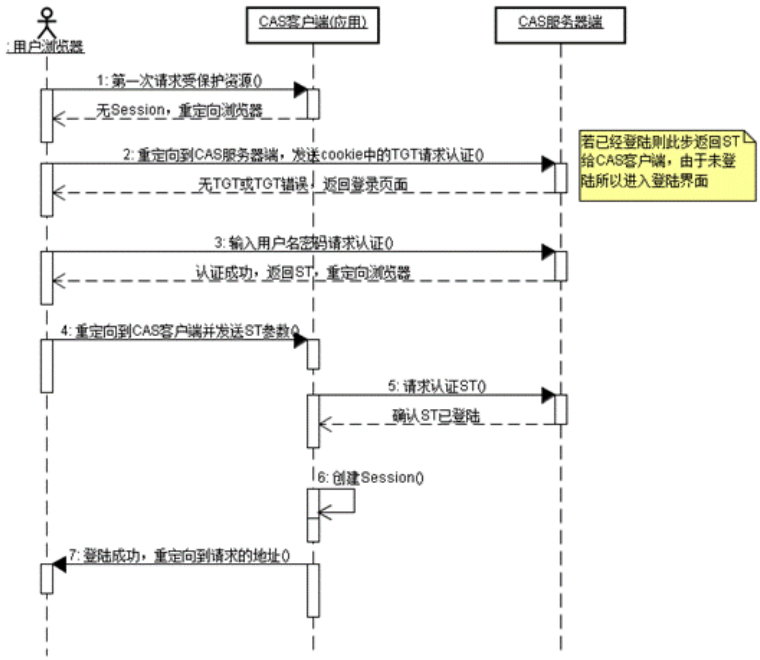


基础协议图

如上图： CAS Client 与受保护的客户端应用部署在一起，以 Filter 方式保护 Web 应用的受保护资源，过滤从客户端过来的每一个 Web 请求，同时， CAS Client 会分析 HTTP 请求中是否包含请求 Service Ticket(ST 上图中的 Ticket)，如果没有，则说明该用户是没有经过认证的；于是 CAS Client 会重定向用户请求到 CAS Server （ Step 2 ），并传递 Service （要访问的目的资源地址）。 Step 3 是用户认证过程，如果用户提供了正确的 Credentials， CAS Server 随机产生一个相当长度、唯一、不可伪造的 Service Ticket，并缓存以待将来验证，并且重定向用户到 Service 所在地址（附带刚才产生的 Service Ticket），**并为客户端浏览器设置一个** Ticket Granted Cookie （ TGC ）； CAS Client 在拿到 Service 和新产生的 Ticket 过后，在 Step 5 和 Step6 中与 CAS Server 进行身份核实，以确保 Service Ticket 的合法性。

在该协议中，所有与 CAS Server 的交互均采用 SSL 协议，以确保 ST 和 TGC 的安全性。协议工作过程中会有 2 **次重定向** 的过程。但是 CAS Client 与 CAS Server 之间进行 Ticket 验证的过程对于用户是透明的（使用 HttpsURLConnection ）。

CAS 请求认证时序图如下：



3.2.1. CAS 如何实现 SSO

当用户访问另一个应用的服务再次被重定向到 CAS Server 的时候， CAS Server 会主动获到这个 TGC cookie，然后做下面的事情：

1) 如果 User 持有 TGC 且其还没失效，那么就走基础协议图的 Step4，达到了 SSO 的效果；

2) 如果 TGC 失效，那么用户还是要重新认证 (走基础协议图的 Step3) 。

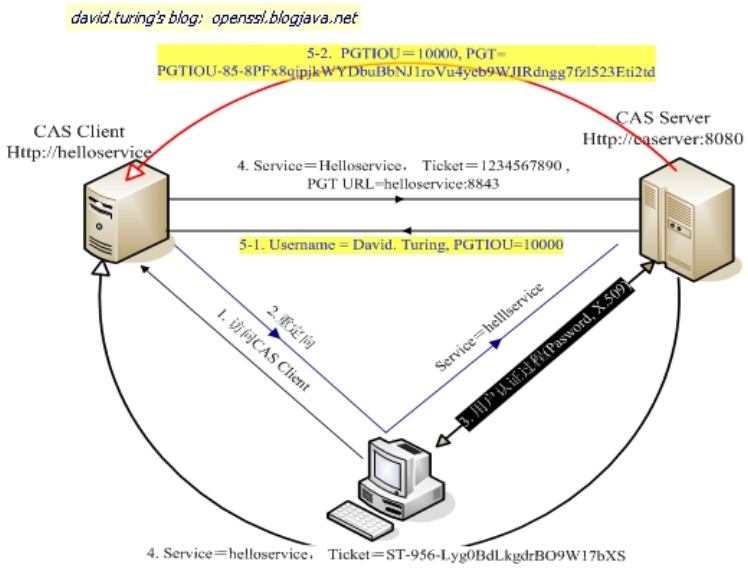
3.2.2. CAS 代理模式

该模式形式为用户访问 App1 ， App1 又依赖于 App2 来获取一些信息，如： User -->App1 -->App 2 。

这种情况下，假设 App2 也是需要对 User 进行身份验证才能访问，那么，为了不影响用户体验（过多的重定向导致 User 的 IE 窗口不停地闪动) ， CAS 引入了一种 Proxy 认证机制，即 CAS Client 可以代理用户去访问其它 Web 应用。

代理的前提是需要 CAS Client 拥有用户的身份信息 (类似凭据) 。之前我们提到的 TGC 是用户持有对自己身份信息的一种凭据，这里的 PGT 就是 CAS Client 端持有的对用户身份信息的一种凭据。凭借 TG C ， User 可以免去输入密码以获取访问其它服务的 Service Ticket ，所以，这里凭借 PGT ， Web 应用可以代理用户去实现后端的认证，而 **无需前端用户的参与** 。

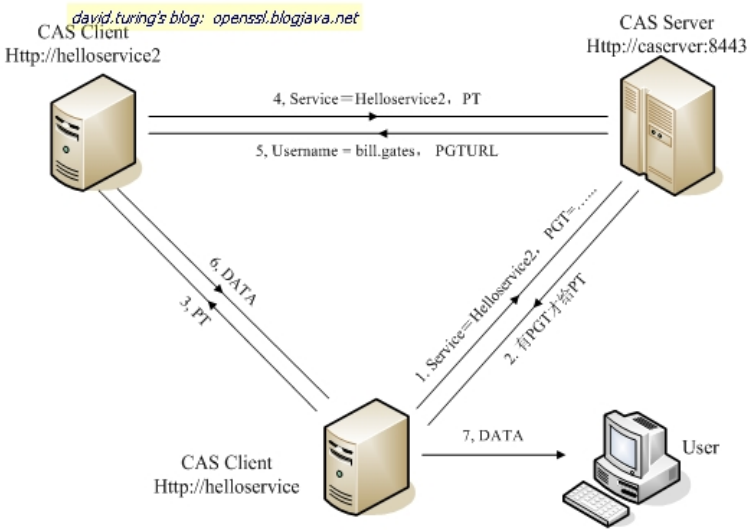
下面为代理应用 （ helloService ） 获取 PGT 的过程： （注： PGTURL 用于表示一个 Proxy 服务，是一个回调链接； PGT 相当于代理证； PGTIOU 为取代理证的钥匙，用来与 PGT 做关联关系；）



如上面的 CAS Proxy 图所示， CAS Client 在基础协议之上，在验证 ST 时提供了一个额外的 PGT URL(而且是 SSL 的入口) 给 CAS Server ，使得 CAS Server 可以通过 PGT URL 提供一个 PGT 给 CA S Client 。

CAS Client 拿到了 PGT(PGTIOU-85 … ..ti2td) ，就可以通过 PGT 向后端 Web 应用进行认证。

下面是代理认证和提供服务的过程：



如上图所示， Proxy 认证与普通的认证其实差别不大， Step1 ， 2 与基础模式的 Step1,2 几乎一样，唯一不同的是， Proxy 模式用的是 PGT 而不是 TGC ，是 Proxy Ticket （ PT ）而不是 Service Ti cket 。

3.2.3. 辅助说明

CAS 的 SSO 实现方式可简化理解为： 1 个 Cookie 和 N 个 Session 。 CAS Server 创建 cookie ，在所有应用认证时使用，各应用通过创建各自的 Session 来标识用户是否已登录。

用户在一个应用验证通过后，以后用户在同一浏览器里访问此应用时，客户端应用中的过滤器会在 session 里读取到用户信息，所以就不会去 CAS Server 认证。如果在此浏览器里访问别的 web 应用时，客户端应用中的过滤器在 session 里读取不到用户信息，就会去 CAS Server 的 login 接口认证，但这时 CAS Server 会读取到浏览器传来的 cookie （ TGC ），所以 CAS Server 不会要求用户去登录页面登录，只是会根据 service 参数生成一个 Ticket ，然后再和 web 应用做一个验证 ticket 的交互而已。

3.3. 术语解释

CAS 系统中设计了 5 中票据： TGC 、 ST 、 PGT 、 PGTIOU 、 PT 。

- Ticket-granting cookie(TGC)： 存放用户身份认证凭证的 cookie ，在浏览器和 CAS Server 间通讯时使用，并且只能基于安全通道传输（ Https ），是 CAS Server 用来明确用户身份的凭证；
- Service ticket(ST)： 服务票据，服务的惟一标识码 , 由 CAS Server 发出（ Http 传送），通过客户端浏览器到达业务服务器端；一个特定的服务只能有一个惟一的 ST ；
- Proxy-Granting ticket （ PGT ）： 由 CAS Server 颁发给拥有 ST 凭证的服务， PGT 绑定一个用户的特定服务，使其拥有向 CAS Server 申请，获得 PT 的能力；
- Proxy-Granting Ticket I Owe You （ PGTIOU ）：作用是将通过凭证校验时的应答信息由 CAS Server 返回给 CAS Client ， 同时，与该 PGTIOU 对应的 PGT 将通过回调链接传给 Web 应用。 Web 应用负责维护 PGTIOU 与 PGT 之间映射关系的内容表；
- Proxy Ticket (PT)： 是应用程序代理用户身份对目标程序进行访问的凭证；

其它说明如下：

- Ticket Granting ticket(TGT)： 票据授权票据，由 KDC 的 AS 发放。即获取这样一张票据后，以后申请各种其他服务票据 (ST) 便不必再向 KDC 提交身份认证信息 (Credentials) ；
- Authentication service(AS) --------- 认证用服务，索取 Credentials ， 发放 TGT ；
- Ticket-granting service (TGS) --------- 票据授权服务，索取 TGT ， 发放 ST ；
- KDC(Key Distribution Center) ---------- 密钥发放中心；

4. CAS 安全性

CAS 的安全性仅仅依赖于 SSL 。使用的是 secure cookie 。

4.1. TGC/PGT 安全性

对于一个 CAS 用户来说，最重要是要保护它的 TGC ， 如果 TGC 不慎被 CAS Server 以外的实体获得， Hacker 能够找到该 TGC ，然后冒充 CAS 用户访问 所有 授权资源。 PGT 的角色跟 TGC 是一样的。

从基础模式可以看出， TGC 是 CAS Server 通过 SSL 方式发送给终端用户，因此，要截取 TGC 难度非常大，从而确保 CAS 的安全性。

TGT 的存活周期默认为 120 分钟。

4.2. ST/PT 安全性

ST （ Service Ticket ）是通过 Http 传送的，因此网络中的其他人可以 Sniffer 到其他人的 Ticket 。 CAS 通过以下几方面来使 ST 变得更加安全（事实上都是可以配置的）：

1、 ST 只能使用一次

CAS 协议规定，无论 Service Ticket 验证是否成功， CAS Server 都会清除服务端缓存中的该 Ticket ，从而可以确保一个 Service Ticket 不被使用两次。

2、 ST 在一段时间内失效

CAS 规定 ST 只能存活一定的时间，然后 CAS Server 会让它失效。默认有效时间为 5 分钟。

3、 ST 是基于随机数生成的

ST 必须足够随机，如果 ST 生成规则被猜出， Hacker 就等于绕过 CAS 认证，直接访问 **对应的** 服务。

5. 参考资料

1、 https://wiki.jasig.org/display/CASUM/Introduction

2、 http://www.jasig.org/cas/protocol/

3、 http://www.ibm.com/developerworks/cn/opensource/os-cn-cas/index.html

4、 http://www.blogjava.net/security/archive/2006/10/02/sso_in_action.html

5、 http://baike.baidu.com/view/190743.htm

上一篇:

下一篇: CAS Ticket票据:TGT、ST、PGT、PT、PGTIOU